Claims

This listing of claims replaces all prior versions and listings:

1. (Currently Amended) A solution for a data file, the data file having a hierarchical arrangement of a plurality of nodes, each said node each having a structure, and wherein the solution comprises:

a ~~special~~ unique name ~~that can be~~ that is the same as a name computed from a solution identifier in the data file~~, wherein the solution can be discovered for deployment by~~ and enables discovery and deployment of the solution ~~using the special name when executed in conjunction with an opening of~~ when the data file is opened; and

a presentation application which, when executed in conjunction with ~~the~~ opening ~~of~~ the data file, displays data in the data file in a plurality of data-entry fields in an electronic form, wherein the plurality of data-entry fields of the electronic form are mapped to a corresponding plurality of the nodes of the data file.

2. (Currently Amended) The solution as defined in Claim 1, wherein the ~~solution for the data file can be discovered and deployed~~ unique name is computed without user interaction and by an electronic forms application executing on a computer ~~that is~~ and configured to discover and deploy the solution of the data file.

3. (Currently Amended) The solution as defined in Claim 1, wherein:

the electronic form is written in XHTML and is rendered when the presentation application is executed by an electronic forms application executing on a computer; and

the plurality of data-entry fields of the electronic form are mapped to a corresponding plurality of the nodes of the data file; and

through each said data-entry field:

data ~~can be~~ is received by input from a user entering the data into the electronic form through one or more of the data-entry fields for storage in a corresponding said node in the data file; and

data in the data file ~~can be~~ is viewed by the user in the electronic form through the data-entry fields via the mapping of the data-entry field from corresponding said nodes of the data file.

4. (Original) The solution as defined in Claim 3, wherein when a logic application is executed by the electronic forms application executing on the computer:

a validation is performed to determine if the data received by input from the user is valid or invalid; and

when the validation determines that the data received by input from the user is invalid, the electronic forms application outputs a dialog box bearing indicia informing the user that the data input is invalid.

5. (Original) The solution as defined in Claim 4, wherein:

the validation of the data received by input from the user into each said data-entry field is performed with a validation rule;

the logic application comprises a plurality of the validation rules for:

a corresponding plurality of the nodes in the data file; and

a corresponding plurality of the data-entry fields;

the validation uses each said validation rule to:

determine if the data received by input from the user into a corresponding said data-entry field is valid or invalid; and

require the user to correct any data input into the corresponding said data-entry field that the validation determines to be invalid.

6. (Original) The solution as defined in Claim 5, wherein each said validation rule is based on a part of a schema governing a corresponding said node.

7. (Original) The solution as defined in Claim 5, wherein each said validation rule is written in script and associated with a corresponding said node.

8. (Original) The solution as defined in Claim 5, wherein each said validation rule is written in a declarative syntax and associated with a corresponding said node.

9. (Original) The solution as defined in Claim 5, wherein:

each said validation rule includes an alert area display; and

when the validation uses one said validation rule to determine that the data received by input from the user into a corresponding said data-entry field is invalid, the corresponding alert area display is output so as to be associated with the corresponding said data-entry field.

10. (Original) The solution as defined in Claim 9, wherein when the alert area display is output, the output includes one or more characteristics selected from the group consisting of:

graphics surrounding the corresponding said data-entry field;

the alert area display surrounds the corresponding said data-entry field;

the alert area display includes graphics containing a red, dashed-lined box;

the alert area display includes graphics highlighting the data in the corresponding said data-entry field;

the alert area display surrounds the corresponding said data-entry field and includes the graphics containing a squiggly line beneath the data in the corresponding said data-entry field;

the alert area display includes text containing information about the invalid data in the corresponding said data-entry field;

the alert area display includes text containing information about the corresponding said data-entry field; and

the alert area display includes a pop-up window.

11. (Original) The solution as defined in Claim 5, wherein each said node has one or more of the validation rules associated therewith.

12. (Original) The solution as defined in Claim 5, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be within a certain range.

13. (Original) The solution as defined in Claim 5, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be within a certain range of text or numerals for a setting of one or more bounds of the certain range.

14. (Original) The solution as defined in Claim 5, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be numerical.

15. (Original) The solution as defined in Claim 5, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be textual.

16. (Original) The solution as defined in Claim 5, wherein one said validation rule includes a requirement for the data received by input from the user into a corresponding said data-entry field that references another said node in the data file.

17. (Original) The solution as defined in Claim 5, wherein the plurality of the validation rules are associated by mapping to the corresponding plurality of the nodes in the data file.

18. (Original) The solution as defined in Claim 5, wherein each said validation rule is associated by mapping to a corresponding said data-entry field by an XPath expression.

19. (Original) The solution as defined in Claim 5, wherein each said validation rule is associated by mapping to a corresponding said data-entry field by use of a declarative syntax.

20. (Original) The solution as defined in Claim 5, wherein each said validation rule is script-based.

21. (Original) The solution as defined in Claim 20, wherein the script-based validation rule maps to a corresponding said node with an XPath expression.

22. (Original) The solution as defined in Claim 20, wherein the script-based validation rule maps to a corresponding said node with an event handler.

23. (Original) The solution as defined in Claim 20, wherein the script-based validation rule maps to a corresponding said node with an event handler that determines when a real-time validation tool uses the script-based validation rule.

24. (Original) The solution as defined in Claim 20, wherein the script-based validation rule maps to a corresponding said node with an event handler that determines when a real-time validation tool uses the script-based validation rule before data received for the node is held by the data file.

25. (Original) The solution as defined in Claim 20, wherein the script-based validation rule maps to a corresponding said node with an event handler that determines when a real-time validation tool uses the script-based validation rule after data received for the node is held by the data file.

26. (Original) The solution as defined in Claim 5, wherein each said validation rule includes:

an alert area display; and

how the alert area display is to appear when output.

27. (Currently Amended) A solution for a data file, the data file having a hierarchical arrangement of a plurality of nodes, each said node each having a structure, and wherein the solution comprises:

a unique name that is the same as a name computed from a solution identifier in the data file and enables discovery and deployment of the solution when the data file is opened;

a presentation application which, when executed in conjunction with the opening of the data file, displays data in the data file in a plurality of data-entry fields in an electronic form, wherein the plurality of data-entry fields of the electronic form are mapped to a corresponding plurality of the nodes of the data file; and

a manifest of all files ~~that can be used~~ usable for:

representing the data file in the electronic form;

allowing a user ~~in~~ to input data into the data-entry fields; and

validating the data that the user inputs into the data-entry fields.

28. (Currently Amended) The solution as defined in Claim 27, wherein:

the data file is written in XML ~~and includes a solution identifier~~;

the presentation application is written in XSLT; and

the electronic form is written in XHTML[[;]]

~~a special name can be computed from a solution identifier in the data file; and~~

~~the solution can be discovered for deployment by using the special name when executed in conjunction with the opening of the data file~~.

29. (Original) A solution for a structured markup-language document, the solution comprising:

a markup-language schema based on the structured markup-language document, wherein fragments of the markup-language schema are coupled with portions of the markup-language document; and

a presentation application which, when executed, transforms the coupled portions of the markup-language document into an electronic form.

30. (Original) The solution as defined in Claim 29, wherein:

the structured markup-language document is written in XML;

the presentation application is written in XSLT;

the electronic form is written in XHTML and has one or more data-entry fields; and

the solution further comprises a manifest of all files that can be used for:

representing the structured markup-language document in the electronic form;

allowing a user in input data into the one or more data-entry fields; and

validating the data that the user inputs into the one or more data-entry fields.

31. (Original) The solution as defined in Claim 29, wherein the coupled portions contain information setting forth all possible structured markup-language documents for the coupled portions.

32. (Original) The solution as defined in Claim 29, further comprising a special name that can be computed from a solution identifier in the markup-language document, wherein the solution can be discovered using the special name.

33. (Original) The solution as defined in Claim 32, wherein the solution identifier is a URL.

34. (Original) The solution as defined in Claim 32, wherein the solution identifier is a URN.

35. (Original) The solution as defined in Claim 32, wherein the special name is unique.

36. (Original) The solution as defined in Claim 32, wherein the special name that can be computed from a solution identifier in the data file is performed with a one-way hash.

37. (Original) The solution as defined in Claim 36, wherein the one-way hash is an MD5 hash.

38. (Original) The solution as defined in Claim 32, wherein the solution identifier in the markup-language document can be read from a processing instruction in the markup-language document.

39. (Currently Amended) A solution comprising:

pieces of markup-language data of a markup-language document, the markup-language data having portions, each said portion having a structure; ~~and~~

a unique name that is the same as a name computed from a solution identifier in the markup-language document and enables discovery and deployment of the solution; and

a presentation application which, when executed responsive to being discovered using the unique name, transforms the pieces of markup-language data into an electronic form according to the structure of the corresponding said portions.

40. (Original) The solution as defined in Claim 39, wherein:

the markup-language data is written in XML;

the presentation application is written in XSLT;

the electronic form is written in XHTML and has one or more data-entry fields; and

the solution further comprises a manifest of all files that can be used for:

representing the markup-language data in the electronic form;

allowing a user in input data into the one or more data-entry fields; and

validating the data that the user inputs into the one or more data-entry fields.

41. (Canceled)

42. (Original) A solution comprising:

a markup-language schema from which a markup-language document can be inferred that has a structure based on the markup-language schema, wherein portions of the markup-language document are logically coupled with fragments of the markup-language schema; and

a presentation application which, when executed, forms an electronic form containing data-entry fields associated with the coupled portions.

43. (Original) The solution as defined in Claim 42, wherein:

the markup-language document is written in XML;

the presentation application is written in XSLT;

the electronic form is written in XHTML; and

the solution further comprises a manifest of all files that can be used for:

representing the markup-language document in the electronic form;

allowing a user in input data into the one or more data-entry fields; and

validating the data that the user inputs into the one or more data-entry fields.

44. (Original) The solution as defined in Claim 42, wherein the coupled portions contain information setting forth all possible markup-language documents for the coupled portions.

45. (Original) The solution as defined in Claim 42, wherein the markup-language schema does not conform to a recognized standard.

46. (Original) The solution as defined in Claim 42, wherein the markup-language schema defines an arbitrary syntax.

47. (Original) The solution as defined in Claim 42, further comprising a special name that can be computed from a solution identifier in the markup-language document, wherein the solution can be discovered using the special name.

48. (Currently Amended) A method comprising:

receiving an instruction to open a markup-language document having a structure and a solution identifier;

computing a ~~special~~ unique name using the solution identifier;

discovering a solution using the ~~special~~ unique name;

opening the markup-language document with the solution, wherein:

the solution includes a presentation application and a markup-language schema;

the markup-language document can be inferred from the markup-language schema; and

portions of the markup-language document are logically coupled with fragments of the markup-language schema;

executing the presentation application to render an electronic form containing data-entry fields associated with the coupled portions.

49. (Original) The method as defined in Claim 48, wherein:

the markup-language document is written in XML;

the presentation application is written in XSLT;

the electronic form is written in XHTML; and

the solution further comprises a manifest of all files that can be used for:

representing the markup-language document in the electronic form;

allowing a user in input data into the one or more data-entry fields; and

validating the data that the user inputs into the one or more data-entry fields.

50. (Original) The method as defined in Claim 48, wherein the coupled portions contain information setting forth all possible markup-language documents for the coupled portions.

51. (Original) The method as defined in 48, wherein the data-entry fields of the electronic form map to a corresponding plurality of nodes of the markup-language document; and the method further comprises:

receiving, through one or more said data-entry fields, data input by a user for storage in a corresponding said node in the markup-language document; and

outputting data in the markup-language for viewing by the user in the electronic form through the data-entry fields via the mapping of the data-entry fields from corresponding said nodes of the markup-language document.

52. (Original) The method as defined in Claim 51, wherein:

the markup-language schema includes a logic application; and

the method further comprises:

executing the logic application to perform a validation to determine if the data received by input from the user is valid or invalid; and

when the validation determines that the data received by input from the user is invalid, outputting a dialog box bearing indicia informing the user that the data input is invalid.

53. (Original) The method as defined in Claim 52, wherein:

the validation is performed on the data received by input from the user into each said data-entry field with a validation rule;

the logic application comprises a plurality of the validation rules for:

a corresponding plurality of the nodes in the markup-language document; and

a corresponding plurality of the data-entry fields;

the validation uses each said validation rule to:

determine if the data received by input from the user into a corresponding said data-entry field is valid or invalid; and

require the user to correct any data input into the corresponding said data-entry field that the validation determines to be invalid.

54. (Original) The method as defined in Claim 53, wherein each said validation rule is based on a part of a schema governing a corresponding said node.

55. (Original) The method as defined in Claim 53, wherein each said validation rule is written in script and associated with a corresponding said node.

56. (Original) The method as defined in Claim 53, wherein each said validation rule is written in a declarative syntax and associated with a corresponding said node.

57. (Original) The method as defined in Claim 53, wherein:

each said validation rule includes an alert area display; and

the validation further comprises using one said validation rule to determine that the data received by input from the user into a corresponding said data-entry field is invalid and outputting the corresponding alert area display so as to be associated with the corresponding said data-entry field.

58. (Original) The method as defined in Claim 57, wherein when the alert area display is output, the output includes one or more characteristics selected from the group consisting of:

graphics surrounding the corresponding said data-entry field;

the alert area display surrounds the corresponding said data-entry field;

the alert area display includes graphics containing a red, dashed-lined box;

the alert area display includes graphics highlighting the data in the corresponding said data-entry field;

the alert area display surrounds the corresponding said data-entry field and includes the graphics containing a squiggly line beneath the data in the corresponding said data-entry field;

the alert area display includes text containing information about the invalid data in the corresponding said data-entry field;

the alert area display includes text containing information about the corresponding said data-entry field; and

the alert area display includes a pop-up window.

59. (Original) The method as defined in Claim 53, wherein each said node has one or more of the validation rules associated therewith.

60. (Original) The method as defined in Claim 53, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be within a certain range.

61. (Original) The method as defined in Claim 53, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be within a certain range of text or numerals for a setting of one or more bounds of the certain range.

62. (Original) The method as defined in Claim 53, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be numerical.

63. (Original) The method as defined in Claim 53, wherein one said validation rule includes a requirement that the data received by input from the user into a corresponding said data-entry field is to be textual.

64. (Original) The method as defined in Claim 53, wherein one said validation rule includes a requirement for the data received by input from the user into a corresponding said data-entry field that references another said node in the data file.

65. (Original) The method as defined in Claim 53, wherein the plurality of the validation rules are associated by mapping to the corresponding plurality of the nodes in the data file.

66. (Original) The method as defined in Claim 53, wherein each said validation rule is associated by mapping to a corresponding said data-entry field by an XPath expression.

67. (Original) The method as defined in Claim 53, wherein each said validation rule is associated by mapping to a corresponding said data-entry field by use of a declarative syntax.

68. (Original) The method as defined in Claim 53, wherein each said validation rule is script-based.

69. (Original) The method as defined in Claim 68, wherein the script-based validation rule maps to a corresponding said node with an XPath expression.

70. (Original) The method as defined in Claim 68, wherein the script-based validation rule maps to a corresponding said node with an event handler.

71. (Original) The method as defined in Claim 68, wherein the script-based validation rule maps to a corresponding said node with an event handler that determines when a real-time validation tool uses the script-based validation rule.

72. (Original) The method as defined in Claim 68, wherein the script-based validation rule maps to a corresponding said node with an event handler that determines when a real-time validation tool uses the script-based validation rule before data received for the node is held by the data file.

73. (Original) The method as defined in Claim 68, wherein the script-based validation rule maps to a corresponding said node with an event handler that determines when a real-time validation tool uses the script-based validation rule after data received for the node is held by the data file.

74. (Original) The method as defined in Claim 53, wherein each said validation rule includes:

an alert area display; and

how the alert area display is to appear when output.

75. (Original) A computer-readable medium comprising instruction that, when executed by a computer, performs the method of Claim 48.

76. (Original) A computer-readable medium including instructions that, when executed by a computer, perform acts comprising:

initiating a request to open a structured markup-language document including a solution identifier;

using the solution identifier to discover a solution that includes a presentation application and a markup-language schema that is based on the structured markup-language document, wherein fragments of the markup-language schema are coupled with portions of the markup-language document; and

executing the presentation application to transform the coupled portions of the markup-language document into an electronic form.

77. (Original) The computer-readable medium as defined in Claim 76, wherein:

the structured markup-language document is written in XML;

the presentation application is written in XSLT;

the electronic form is written in XHTML and has one or more data-entry fields; and

the solution further comprises a manifest of all files that can be used for:

representing the structured markup-language document in the electronic form;

allowing a user in input data into the one or more data-entry fields; and

validating the data that the user inputs into the one or more data-entry fields.

78. (Original) The computer-readable medium as defined in Claim 76, wherein the coupled portions contain information setting forth all possible structured markup-language documents for the coupled portions.

79. (Original) A computer-readable medium including instructions that, when executed by a computer, perform acts comprising:

initiating a request to open a markup-language document including a solution identifier;

using the solution identifier to discover a solution that includes a presentation application and a markup-language schema from which a markup-language document can be inferred that has a structure based on the markup-language schema, wherein portions of the markup-language document are logically coupled with fragments of the markup-language schema; and

executing the presentation application to form an electronic form containing data-entry fields associated with the coupled portions.

80. (Original) The computer-readable medium as defined in Claim 79, wherein:

the markup-language document is written in XML;

the presentation application is written in XSLT;

the electronic form is written in XHTML and has one or more data-entry fields; and

the solution further comprises a manifest of all files that can be used for:

representing the markup-language document in the electronic form;

allowing a user in input data into the one or more data-entry fields; and

validating the data that the user inputs into the one or more data-entry fields.

81. (Original) The computer-readable medium as defined in Claim 79, wherein the coupled portions contain information setting forth all possible markup-language documents for the coupled portions.